7813

Postel

Network Working Group
Request for Comments #265
NIC 7813
Categories: D.4, D.5, and D.7

Obsoletes: 172

17 November 1971
Abhay Bhushan, MIT
Bob Braden, UCLA
Will Crowther, BBN
Eric Harslem, Rand
John Heafner, Rand
Alex McKenzie, BBN
John Melvin, SRI
Bob Sundberg, Harvard
Dick Watson, SRI
Jim White, UCSB

Rcvd 4JAN72

THE FILE TRANSFER PROTOCOL

This paper is a revision of RFC 172, NIC 6794. The changes to RFC 172 are given below. The protocol is then restated for your convenience.

CHANGES TO RFC 172

1) Two new file transfer requests have been added. These are 'create' and 'append_with_create'.

2) The op code assignments in control transactions have been changed to include the above requests.

3) Two new error codes indicating 'incorrect or missing identifier' and 'file already exists' have been added. New error code assignments reflect this change.

4) Editorial changes to clarify specifications.

I. INTRODUCTION

The file transfer protocol (FTP) is a user-level protocol for file transfer between host computers (including terminal IMPs), on the ARPA computer network (ARPANET). The primary function of FTP is to facilitate transfer of files between hosts and to allow convenient use of storage and file handling capabilities of remote hosts. FTP uses the Data Transfer Protocol described in RFC 264 to achieve transfer of data. This paper assumes knowledge of RFC 264.

The objectives of FTP are to promote sharing of files (computer programs and/or data), encourage implicit (without explicit login) use of computers, and shield the user from variations in file and storage systems of different hosts. These objectives are achieved by specifying a standard file transfer socket and initial connection protocol for implicit use, and using standard conventions for file transfer and related operations.

II. DISCUSSION

A file is considered here to be an ordered set of arbitrary length, consisting of computer data (including programs). Files are uniquely identified in a system by their pathnames. A pathname is (loosely) defined to be the data string which must be input to the file system by a network user in order to identify a file. Pathname usually contains device and/or directory names, and file name. FTP specifications provide standard file system commands, but do not provide standard naming convention at this time. Each user must follow the naming convention of the file system he wishes to use. FTP may be extended later to include standard conventions for pathname structures.

A file may or may not have access controls associated with it. The access controls designate users' access privileges. In absence of access controls, files cannot be protected from accidental or unauthorized usage. It is the prerogative of a serving file system to provide protection, and selective access. FTP provides identifier and password mechanisms for exchange of access control information. It should however be noted, that for file sharing, it is necessary that a user be allowed (subject to access controls) to access files not created by him.

FTP does not restrict the nature of information in files. For example, a file could contain ASCII text, binary data, computer program, or any other information. A provision for indicating data structure (type and byte size) exists in FTP to aid in parsing, interpretation, and storage of data.

To facilitate implicit usage, a serving file transfer process may be a disowned "demon" process which "listens" to an agreed-upon socket, and follows the standard initial connection protocol for establishing a full-duplex connection. It should be noted that FTP may also be used directly by logging into a remote host, and arranging for file transfer over specific sockets.

FTP is readily extendable, in that additional commands and data types may be defined by those agreeing to implement them. Implementation of a subset of commands is specifically permitted, and an initial subset for implementation is recommended. (*)The protocol may also be extended to enable remote execution of programs, but no standard procedure is suggested.

For transferring data, FTP uses the data transfer protocol specified in RFC 264. As the data transfer protocol does not specify the manner in which it is to be used by FTP, implementation may vary at different host sites. Hosts not wishing to separate data transfer and file transfer functions, should take particular care in conforming to the data transfer protocol specifications of RFC 264.

It should be noted that FTP specifications do not require knowledge of transfer modes used by data transfer protocol. However, as file transfer protocol requires the transfer of more than a single control transaction over the same connection, it is essential that hosts be able to send control transactions in either 'transparent block' (type B9) or 'descriptor and counts' (type BA) modes. (Type B8, the indefinite bit stream mode is not suitable as it limits transfer to single transactions.).

The use of data transfer aborts (type B6) is neither required, nor defined in FTP. FTP has its own error terminate which may be used to abort a file transfer request. FTP also does not define the structure of files, and there are no conventions on the use of group, record and unit separators. (*)A file separator however, indicates the end of a file.

It is strongly recommended that default options be provided in implementation to facilitate use of file transfer service. For example, the main file directory on disk, a pool directory, user directory, or directory last accessed could serve as

---

(*)
This initial subset represents control functions necessary for basic file transfer and "mail" operations, and some elementary file manipulation operations. There is no attempt to provide a data management or complete file management capability.

(*)
It is possible that we may, at a later date, assign meaning to these information separators within FTP.

standard pathname defaults. Default mechanisms are convenient, as the user doesn't have to specify the complete pathname each time he wishes to use the file transfer service. No standard default procedures are specified by FTP.

III. SPECIFICATIONS

1. Data Transfer

FTP uses the Data Transfer Protocol (described in RFC 264) for transferring data and/or control transaction. Both data and control transactions are communicated over the same connection.

2. Data Transactions

Data transactions represent the data contained in a file. There is no data type or byte size information contained in data transactions. The structure of data is communicated via control transactions. A file may be transferred as one or more data transactions. The protocol neither specifies nor imposes any limitations on the structure (record, group, etc) or length of file. Such limitations may however be imposed by a serving host. The end of a file may be indicated by a file separator (as defined in data transfer protocol). In the special case of indefinite bit-stream transfer mode (Type BØ), the end of file is indicated by closing connection. In particular, a serving or using host should not send the ETX, or other end of file character, unless such a character is part of the data in file (i.e., not provided by system).

3. Control Transactions

The control transactions may be typified as requests, identifiers, and terminates. A request fulfillment sequence begins with a request and ends with receipt of data (followed by End-of-File) or a terminate. The user side initiates the connection as well as the request. The server side "listens" and complies with the request.

3A. Op Codes

The first information (i.e., not descriptor) byte of control transactions indicates the control function. This byte is referred to as "opcode". A standard set of opcodes are defined below. The operations are discussed in Section 3B.2.

Implementation of a workable subset(*)of opcodes is specifically permitted. Additional standard opcodes may be assigned later. Opcodes hex 5A (octal 1ØØ) through hex FF (octal 377) are for experimental use.

Op Code Hex	Op Code Octal	Operation
ØØ	ØØØ	Set data type identifier
Ø1	ØØ1	Retrieve Request
Ø2	ØØ2	Create request (write file; error if file already exists)
Ø3	ØØ3	Store request (write file; replace if file already exists)
Ø4	ØØ4	Append request (add to existing file; error if file does not exist)
Ø5	ØØ5	Append_with_create request (add to file; create if file does not exist)
Ø6	ØØ6	Delete request (delete file)
Ø7	ØØ7	Rename_from request (change file name)
Ø8	Ø1Ø	Rename_to request (the new file name)
Ø9	Ø11	List request (list information)
ØA	Ø12	Username identifier (for access control)
ØB	Ø13	Password identifier (for access control)
ØC	Ø14	Error or unsuccessful terminate
ØD	Ø15	Acknowledge or successful terminate
ØE through 4F	Ø16 through Ø77	Reserved for standard assignment
5A through FF	1ØØ through 377	Assigned for experimental use

(*) A workable subset is any request, plus terminates. Identifiers may be required in addition for using "protected"

3B. Syntax and Semantics

3B.1 Data Types

The 'set data type' control transactions identifies the structure of data (data type and byte size) in succeeding data transactions. The 'set data type' transaction shall contain two more bytes in addition to the opcode byte. The first of these bytes shall convey a data type or code information and the second byte may convey the data byte size, where applicable. This information may be used to define the manner in which data is to be parsed, interpreted, reconfigured or stored. Set data type need be sent only when structure of data is changed from the preceding.

Although, a number of data types are defined, specific implementations may handle only limited data types or completely ignore the data type and byte size descriptors. Even if a host process does not "recognize" a data type, it must accept data (i.e., there is no such thing as data type error.) These descriptors are provided only for convenience, and it is not essential that they be used. The standard default is to assume nothing about the information and treat it as a bit stream (binary data, byte size 1)(*)whose interpretation is left to a higher level process, or the user.

The following data type codes are currently assigned. Where a byte size is not implicit in data type, it may be provided by the second byte.

---

file systems.

(*)
It is, however, possible that this bit stream is treated like ASCII characters in specific instances such as transmitting a file to a line printer.

Code Hex	Code Octal	Implicit Byte Size	Data Type
ØØ	ØØØ	1	Bit stream (standard default)
Ø1	ØØ1	none	Binary data bytes
Ø2	ØØ2	8	Network ASCII characters
Ø3	ØØ3	8	EBCDIC characters
Ø4	ØØ4	36	DEC-packed ASCII (five 7-bit characters, 36th bit 1 or Ø)
Ø5	ØØ5	8	Decimal numbers, net. ASCII
Ø6	ØØ6	8	Octal numbers, net. ASCII
Ø7	ØØ7	8	Hexadecimal numbers, net. ASCII
Ø8 through 4F	Ø1Ø through Ø77		Reserved for standard assignment
5Ø through FF	1ØØ through 377		Assigned for experimental use

3B.2 Requests and Identifiers

Retrieve, create, append, append_with_create, delete, rename_from, and rename_to requests must contain a pathname specifying a file, following the opcode in the information field. In the list request a pathname may or may not follow the opcode. If present, the pathname may specify either a file or a directory.

A file pathname must uniquely identify a file in the serving host. The syntax of pathnames and identifying information shall conform to serving host conventions, except that standard network ASCII (7-bit ASCII right justified in 8-bit field with most significant bit as zero) shall be used.

The store request has a 4-byte (32 bits) 'alloctate size' field followed by a pathname specifying a file. 'Allocate size' indicates the number of bits of storage to be allocated to the file. An allocate size of zero indicates that server should use his default.

Retrieve request achieves the transfer of a copy of file specified in pathname, from serving to using host. The

status and contents of file in serving host should be unaffected.

Create request causes a file to be created at the serving host as specified in pathname. A copy of the file is transferred from the using to the serving host. If the file specified in pathname already exists at the serving host, an error terminate should be sent by the server.

Store request achieves the transfer of a copy of file from using to serving host. If file specified in pathname exists on serving hosts, then its contents shall be replaced by the contents of the file being transferred. A new file is created at the serving host if the file specified in pathname does not exist.

Append request achieves the transfer of data from using to serving host. The transferred data is appended to file specified in pathname, at serving host. If the specified file does not exist at serving host, an error terminate should be sent by the server.

Append with create request achieves the transfer of data from using to serving host. If file specified in pathname exists at serving host, then the transferred data is appended to that file, otherwise the file specified in pathname is created at the serving host.

Rename from and rename to requests cause the name of the file specified in pathname of rename_from to be changed to the name specified in pathname of rename_to. A rename_from request must always be followed by a rename_to request.

Delete request causes file specified in pathname to be deleted from the serving host. If an extra level of protection is desired such as the query "Do you really wish to delete this file?", it is to be a local implementation option in the using system. Such queries should not be transmitted over network connections.

List request causes a list to be sent from the serving to using host. If there is no pathname or if pathname is a directory, the server should send a file directory list. If the pathname specifies a file then server should send current information on the file.

Username and password identifiers contain the respective identifying information. Normally, the information will be supplied by the user of the file transfer service. These identifiers will normally be sent at the start of connection for access control.

5B.3 Error and Acknowledge Terminates

The error transactions may have an error code indicated by the second information byte. Transmission of an ASCII error message in subsequent bytes is permitted with all error codes, except that with Hex'ØA' error code, ASCII text is required. The errors here relate to file transfer functions only. Data synchronization and related errors in data transfer are to be handled at the DTP level. The following error codes are currently defined:

Error Code (2nd descriptor byte)		Meaning
Hex	Octal	
ØØ	ØØØ	Error condition indicated by computer system (external to protocol)
Ø1	ØØ1	Name syntax error
Ø2	ØØ2	Access control violation
Ø3	ØØ3	Abort (by user)
Ø4	ØØ4	Allocate size too big
Ø5	ØØ5	Allocate size overflow
Ø6	ØØ6	Improper order for transactions
Ø7	ØØ7	Opcode not implemented
Ø8	Ø1Ø	File search failed
Ø9	Ø11	Incorrect or missing identifier
ØA	Ø12	Error described in text message (ASCII characters follow code)
ØB	Ø13	File already exists (in create request)

At present, no completion codes are defined for acknowledge. It is assumed that acknowledge refers to the current request being fulfilled.

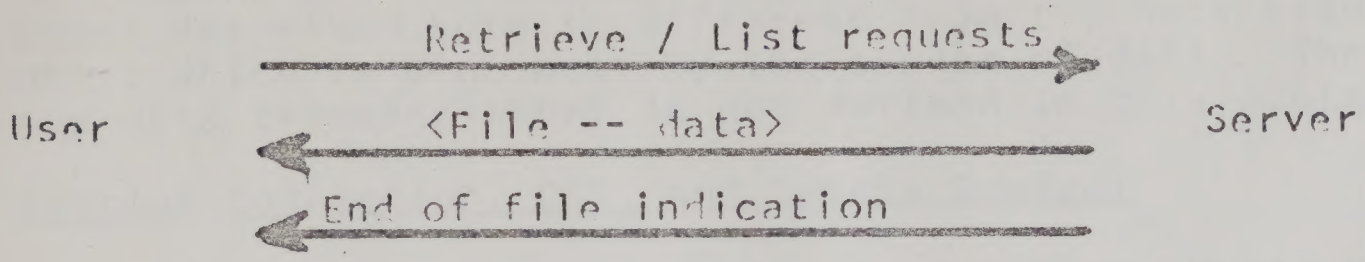
4. Order of Transactions

4A. A certain order of transactions must be maintained in fulfilling file transfer requests. The exact sequence in which transactions occur depends on the type of request, as described in section 4B. The fulfillment of a request may be aborted anytime by either host, as explained in section 4C.

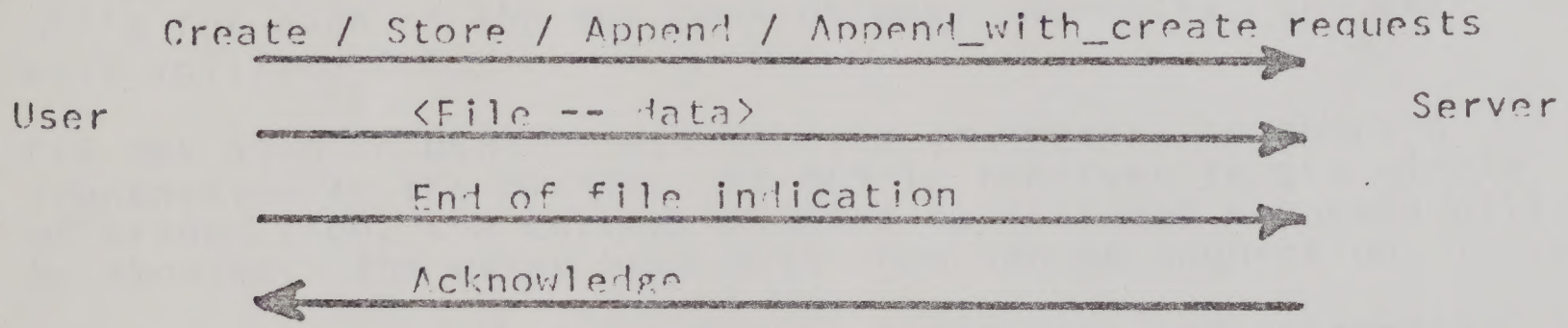
4B. Identifier transactions (set data type, username, and password) may be sent by user at any time. The usual order would be a username transaction followed by a password transaction at the start of the connection. No acknowledge is required, or permitted. The identifiers are to be used for default handling, and access control.

Retrieve and list requests cause the transfer of file from server to user. After a complete file has been transferred, the server should indicate end-of-file (by sending CLS or

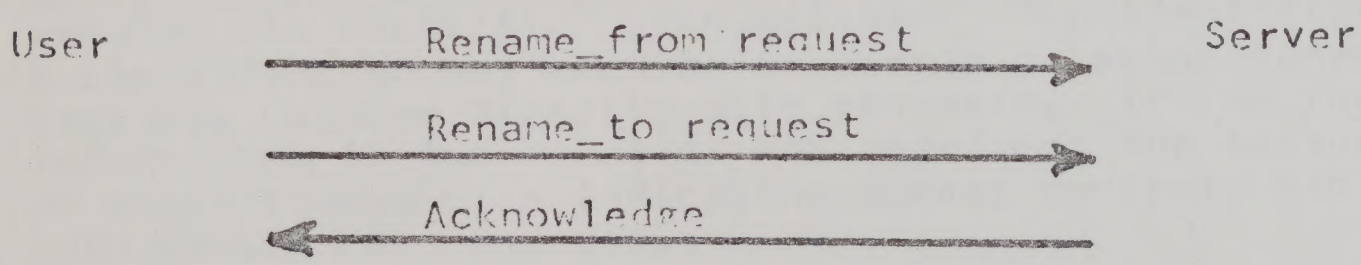
file separator) to complete the request fulfillment sequence, as shown below.



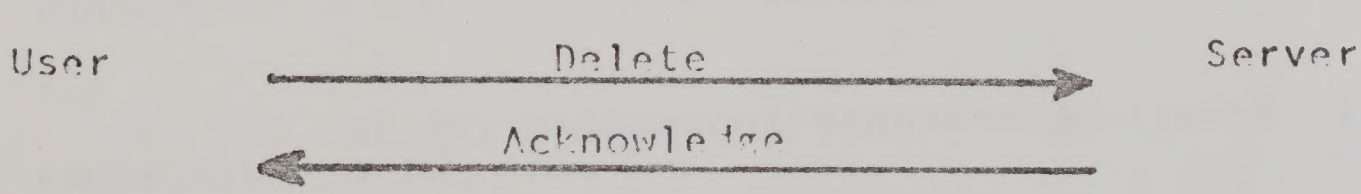
Store, create, append, and append_with_create requests cause the transfer of file from user to server. After a complete file has been transferred, the user should send an end-of-file indication. The receipt of the file must be acknowledged by the server, as shown below.



Rename_from request must be followed by a rename_to request. The request must be acknowledged as shown below.



The delete request requires the server to acknowledge it, as shown below.



Error transactions may be sent by either host at any time, and these terminate the current request fulfillment sequence.

4C. Aborts. Either host may abort a request fulfillment sequence at any time by sending an error terminate, or by closing the connection (NCP to transmit a CLS for the connection). CLS is a more drastic type of abort and shall be used when there is a catastrophic failure, or when abort is desired in the middle of a long transaction. The abort indicates to the

receiving host that sender of abort wishes to terminate request fulfillment and is now ready to initiate or fulfill new requests. When CLS is used to abort, the using host will be responsible for reopening connection. The file transfer abort described here is different from the data transfer abort which is sent only by the sender of data. The use of the data transfer abort is not defined in this protocol.

5. Initial Connection, CLS, and Access Control

5A. Socket 3 is the standard preassigned socket number on which the cooperating file transfer process at the serving host should "listen". (*)The connection establishment will be in accordance with the standard initial connection protocol,(*)establishing a full-duplex connection.

5B. The connection will be broken by trading a CLS between the NCP's for each of the two connections. Normally, the user will initiate CLS.

CLS may also be used by either user or server, to abort a transaction in the middle. If CLS is received in the middle of transaction, the current request fulfillment sequence will be aborted. The using host will then reopen connection.

5C. It is recommended that identifier (user name and password) transactions be sent by user to server, at the start, as this would facilitate default handling and access control for the entire duration of connection. Some service sites may require the identifier transactions. The identifier transactions do not require or permit an acknowledge, and the user can proceed directly with requests. If the identifier information is incorrect or not received, the server may send an error transaction indicating access control, violation, upon subsequent requests.

(*)
Socket 1 has been assigned to logger, socket 3 seems a reasonable choice for File Transfer.

(*)
RFC 165, or any subsequent standard applicable in initial connection to loggers.